Animation Project in C++ using STL Container Classes and Overloaded Operators

Due Time: 23.59, Sat 9 December 2017 **Earnings:** 9% of your final grade

NOTE: Plan to finish a few days early to avoid last minute hardware/software holdups for which no allowance is given.
NOTE: The code in this assignment must be your own work. It must not be code taken from another student or written for you by someone else, even if you give a reference to the person you got it from (attribution); if it is not entirely your own work it will be treated as plagiarism and given a fail mark, or less.

Purpose: This is a development of assignment 1 with the addition of a two ready-made container class templates and overloaded operators. It works in a very similar way. It is a console application that holds the Frames of an Animation application (there is no actual animation graphics in the assignment) using a `forward_list` class template of unspecified length in dynamic memory. The `forward_list` is a container class that supports fast insertion and removal of elements and is implemented as a singly-linked list just as you have used in assignments 0 and 1.
In addition a Frame now has set of Display objects associated with it. You can think of them as the separate graphical components of a Frame that combine the make the complete image (– they might be green/blue-screen overlays as in chroma keying). A Display object holds the pixel location that it is to be overlayed in the frame and also the duration for which it should be displayed. The set of Display objects is held in a `vector` - another common container class template, more flexible but not (for some applications) as fast the forward list, which is designed to be minimal.
Part of the code is shown on the next page; it is also on Blackboard in a text file that you can copy and paste. You **MUST** use this code **without modification (not a single character changed): no code added or removed, no new global variables or functions, no new classes, no macros, no defines and no statics**. Your task is to implement, using C++, only the Animation, Frame and Display class member functions and the global insertion operators and not add any new ones. Everything you write and submit is in the files: Animation.cpp, Frame.cpp and Display.cpp.

The Animation is a series of Frames held in a `forward_list`. Each Frame holds its list of Display objects in a `vector`. A Display object contains its display time which is set by the user - therefore the fixed Frame display time of 1 second that was used in previous assignments is gone and is replaced by the individual Display times that the user specifies. You can:

- Add a new Frame to the Animation at a position in the forward list selected by the user
- Delete all the Frames in the Animation
- Run the Animation to show the list of Display details of each Frame one after another at the display intervals specified by the user when the Display was entered – note that the output counts up the seconds
- concatenate two Frames to make a single Frame with combined Displays that is automatically added to the start of the `forward_list`
- Quit

An example of the output of the running application is given at the end. Yours must work identically and produce identical output.

Note the following:

- dynamic memory management is done with new and delete (not malloc, realloc or free)
- input and output is done with cin and cout
- there is no unused dynamic memory at any time
- string objects are used in the Animation and Frame classes to hold strings (a char array is still used in the Display class)
- Release of dynamically allocated memory is done in destructors so there is no resource leak (or you lose 30%).

See the Marking Sheet for how you can lose marks, but you will lose at least 60% if:

1. you change the supplied code in any way at all (not a single character) - no code added or removed, no macros, no defines, no statics and no additional classes, global functions or variables,
2. it fails to build in Visual Studio 2015,
3. It crashes in normal operation,
4. it doesn't produce the example output.

Part of the code is shown on the next page. You MUST use this code **without modification.** Your task is to add the implementation of the class member functions.

<u>What to Submit :</u> Use Blackboard to submit this assignment as a plain zip file (**not** RAR or 7-Zip or 9 Zip) containing only Animation.cpp, Frame.cpp and Display.cpp. The name of the zipped folder **<u>must</u>** contain your name as a prefix so that I can identify it, for example using my name the file would be tyleraAss2CST8219.zip. It is also vital that you include the Cover Information (as specified in the Submission Standard) as a file header in your source file so the file can be identified as yours. Use comment lines in the file to include the header.

Before you submit the code,

- check that it builds and executes in Visual Studio 2015 as you expect - if it doesn't build for me, for whatever reason, you get a deduction of at least 60%.
- make sure you have submitted the correct file – if I cannot build it because the file is wrong or missing from the zip, even if it's an honest mistake, you get 0.

There is a late penalty of 25% per day. Don't send me files as an email attachment – they will get 0.

Supplied code (also in a text file you can copy and paste on BlackBoard). Don't change it.

```
//Animation.h
#pragma once

class Animation
{
   string name;
   forward_list<Frame> frames;
public:
   Animation(string s): name(s){}
   void InsertFrame();
   void DeleteFrames();
   void Concatenate() // inline
   {
          cout << "Concatenating two Frames" << endl;
          int index1{ -1 }, index2{ -1 }; // same as int index1 = -1, index2 = -1;
          Animation& A{ *this };        // same as Animation& A = *this;
          int count{ distance(frames.begin(),frames.end()) };//same as int count = distance(frames.begin(),frames.end());
          do {
                    cout << "Please enter valid indexes of the two Frames to concatenate (0 to " << count - 1 << ")" << endl;
                    cin >> index1 >> index2;
          } while ((index1<0 || index1>count - 1) || (index2<0 || index2>count - 1));
          A += A[index1] + A[index2];
   }
   Frame& operator[](unsigned int);
   void operator+=(Frame&);
   friend ostream& operator<<(ostream& , Animation&);
};
```

```
// Display.h
#pragma once

class Display
{
          int pixel_x;
          int pixel_y;
          int duration;
          char* name;
public:
          Display(int x, int y, int duration, char* name);
          Display(const Display&);
          ~Display();
          friend ostream& operator<<(ostream&, Display&);
};
```

```
// Frame.h
#pragma once

class Frame
{
          string fileName;
          vector<Display> displays;
public:
          Frame::Frame(string s, vector<Display> d) :fileName(s), displays(d){}
          Frame operator+(Frame&);
          friend ostream& operator<<(ostream&, Frame&);
};
```

```
// ass2.cpp
#define _CRT_SECURE_NO_WARNINGS
#define _CRTDBG_MAP_ALLOC     // need this to get the line identification
//_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF|_CRTDBG_LEAK_CHECK_DF); // in main, after local declarations
//NB must be in debug build

#include <crtdbg.h>
#include <iostream>
#include <string>
#include <vector>
#include <forward_list>
using namespace std;

#include "Display.h"
#include "Frame.h"
#include "Animation.h"

bool running = true;

int main(void)
{
          char selection;
          bool running = true;
          Animation A("A");
          _CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF); // in main, after local declarations


          while (running)
          {
                    cout<<    "MENU\n 1. Insert a Frame\n 2. Delete all the Frames\n 3. Concatenate two Frames\n 4. Run the
Animation\n 5. Quit\n"<<endl;
                    cin >> selection;
                    switch (selection)
                    {
                    case '1':
                              A.InsertFrame();
                              break;
                    case '2':
                              A.DeleteFrames();
                              break;
                    case '3':
                              A.Concatenate();
                              break;
                    case '4':
                              cout << A << endl;
                              break;
                    case '5':
                              running = false;
                              break;
                    default:
                              break;
                    }
          }

          return 0;
}
```

Example Output

```
MENU
 1. Insert a Frame
 2. Delete all the Frames
 3. Concatenate two Frames
 4. Run the Animation
 5. Quit

1
Insert a Frame in the Animation
Please enter the Frame filename: Animation1
Entering the Frame Displays (the sets of dimensions and durations)
Please enter the number of Displays: 2
Please enter pixel x for Display #0 pixel_x:0
Please enter pixel y for Display #0 pixel_y:0
Please enter the duration sec for this Display: 2
Please enter the name for this Display: Display_01
Please enter pixel x for Display #1 pixel_x:1
Please enter pixel y for Display #1 pixel_y:10
Please enter the duration sec for this Display: 3
Please enter the name for this Display: Display_02
This is the first Frame in the list
```

```
MENU
 1. Insert a Frame
 2. Delete all the Frames
 3. Concatenate two Frames
 4. Run the Animation
 5. Quit

1
Insert a Frame in the Animation
Please enter the Frame filename: Animation2
Entering the Frame Displays (the sets of dimensions and durations)
Please enter the number of Displays: 3
Please enter pixel x for Display #0 pixel_x:16
Please enter pixel y for Display #0 pixel_y:32
Please enter the duration sec for this Display: 3
Please enter the name for this Display: Display_01
Please enter pixel x for Display #1 pixel_x:32
Please enter pixel y for Display #1 pixel_y:16
Please enter the duration sec for this Display: 1
Please enter the name for this Display: Display_02
Please enter pixel x for Display #2 pixel_x:128
Please enter pixel y for Display #2 pixel_y:128
Please enter the duration sec for this Display: 4
Please enter the name for this Display: Display_03

MENU
 1. Insert a Frame
 2. Delete all the Frames
 3. Concatenate two Frames
 4. Run the Animation
 5. Quit

1
Insert a Frame in the Animation
Please enter the Frame filename: Animation3
Entering the Frame Displays (the sets of dimensions and durations)
Please enter the number of Displays: 1
Please enter pixel x for Display #0 pixel_x:256
Please enter pixel y for Display #0 pixel_y:412
Please enter the duration sec for this Display: 1
Please enter the name for this Display: Display_01
There are 2 Frame(s) in the list
Please specify the position, between 0 and 1 to insert after : 0

MENU
 1. Insert a Frame
 2. Delete all the Frames
 3. Concatenate two Frames
 4. Run the Animation
 5. Quit

4
Animation A
Run the Animation
Frame #0:
        fileName = Animation1
        Display #0:     name = Display_01; pixel_x = 0; pixel_y = 0; duration = 2
        Counting the seconds for this Display: 1, 2,
        Display #1:     name = Display_02; pixel_x = 1; pixel_y = 10; duration = 3
        Counting the seconds for this Display: 1, 2, 3,
Frame #1:
        fileName = Animation3
        Display #0:     name = Display_01; pixel_x = 256; pixel_y = 412; duration = 1
        Counting the seconds for this Display: 1,
Frame #2:
        fileName = Animation2
        Display #0:     name = Display_01; pixel_x = 16; pixel_y = 32; duration = 3
        Counting the seconds for this Display: 1, 2, 3,
        Display #1:     name = Display_02; pixel_x = 32; pixel_y = 16; duration = 1
        Counting the seconds for this Display: 1,
        Display #2:     name = Display_03; pixel_x = 128; pixel_y = 128; duration = 4
        Counting the seconds for this Display: 1, 2, 3, 4,

Output finished

MENU
 1. Insert a Frame
 2. Delete all the Frames
 3. Concatenate two Frames
 4. Run the Animation
 5. Quit

3
Concatenating two Frames
Please enter valid indexes of the two Frames to concatenate (0 to 2)
1
0

MENU
 1. Insert a Frame
 2. Delete all the Frames
 3. Concatenate two Frames
 4. Run the Animation
 5. Quit
```

```
4
Animation A
Run the Animation
Frame #0:
        fileName = Animation3_Animation1
        Display #0:     name = Display_01; pixel_x = 256; pixel_y = 412; duration = 1
        Counting the seconds for this Display: 1,
        Display #1:     name = Display_01; pixel_x = 0; pixel_y = 0; duration = 2
        Counting the seconds for this Display: 1, 2,
        Display #2:     name = Display_02; pixel_x = 1; pixel_y = 10; duration = 3
        Counting the seconds for this Display: 1, 2, 3,
Frame #1:
        fileName = Animation1
        Display #0:     name = Display_01; pixel_x = 0; pixel_y = 0; duration = 2
        Counting the seconds for this Display: 1, 2,
        Display #1:     name = Display_02; pixel_x = 1; pixel_y = 10; duration = 3
        Counting the seconds for this Display: 1, 2, 3,
Frame #2:
        fileName = Animation3
        Display #0:     name = Display_01; pixel_x = 256; pixel_y = 412; duration = 1
        Counting the seconds for this Display: 1,
Frame #3:
        fileName = Animation2
        Display #0:     name = Display_01; pixel_x = 16; pixel_y = 32; duration = 3
        Counting the seconds for this Display: 1, 2, 3,
        Display #1:     name = Display_02; pixel_x = 32; pixel_y = 16; duration = 1
        Counting the seconds for this Display: 1,
        Display #2:     name = Display_03; pixel_x = 128; pixel_y = 128; duration = 4
        Counting the seconds for this Display: 1, 2, 3, 4,

Output finished

MENU
 1. Insert a Frame
 2. Delete all the Frames
 3. Concatenate two Frames
 4. Run the Animation
 5. Quit
```